Analysis and modeling of ELM stability in DIII-D experiments with OMFIT

BOUT++ workshop - LLNL

Orso Meneghini

P. Snyder, S. Smith, L. Lao, J. Candy, C. Holland, O. Izacard, T. Osborne, R. Prater, H. St John, A. Turnbull

Oak Ridge Associated Universities
General Atomics

Sept 5^{th} 2013

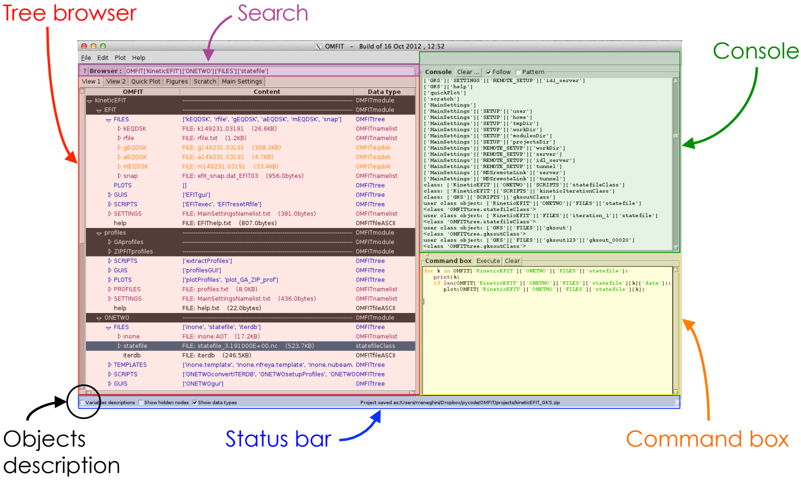
Oak Ridge Institute for Science and Education

Outline

Outline

One Modeling Framework for Integrated Tasks (OMFIT)

A framework for the every-day analysis and modeling needs of both theorists and experimentalist!



One Modeling Framework for Integrated Tasks (OMFIT)

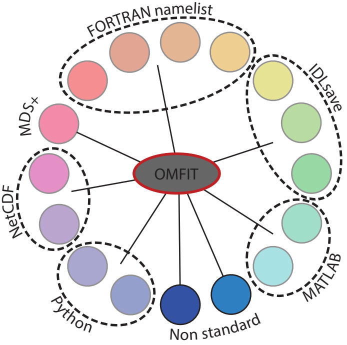
1. a **workflow manager**
   - Data "flows" through different physics components
   - Not a *transport solver*... that is just another component
2. for **shallow code integration**
   - Stand-alone codes share "small" quantities of data
   - I/O of stand-alone codes is mostly done by files
3. following a **BOTTOM-UP, grassroots approach:**
   - Framework provides the tools for creating, improving, integrating components
   - Users decide what codes to couple and how they interact
   - Sharing of modules and their improvements
     Grows depending on the most pressing interest of the community
     Example: encyclopedia **vs.** Wikipedia

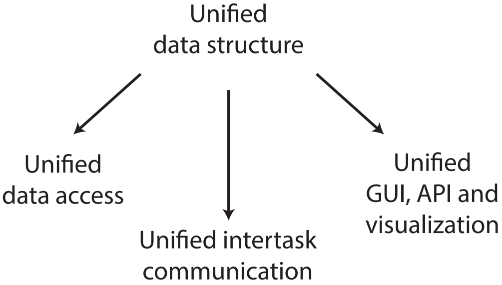
OMFIT philosophy and design choices

- **Recognize and encourage reuse of existing work**
  - Use any file formats
  - Integrate existing scripts/widgets/softwares
- **Ease the way of working...**
  - Interactive graphical environment
  - High level API
  - Quick visualization of data
- **...without limiting possibilities**
  - User-level scripting to drive workflow
  - Freedom to organize data as necessary
  - All output data / input parameters always accessible
- **From experimental data to data analysis and modeling**
  - Integration with experimental databases
- **Create a cooperative environment**
  - Sharing of knowledge among users
  - Open-source

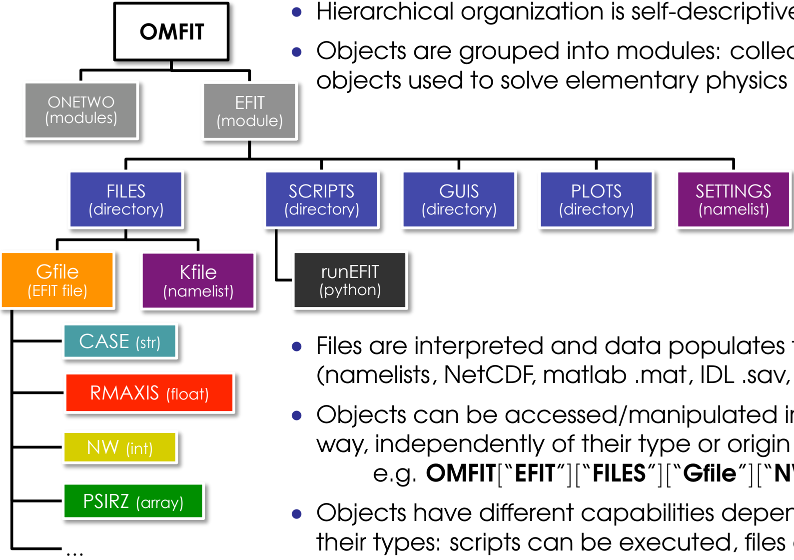
Main idea: to treat files, scripts, experiment data, texts, plots, executable, ... as a uniform collection of objects

- Centralize data from different sources
- Store everything deemed relevant, with no a-priory decision of what is stored and how
- Read/write of relatively few scientific data formats makes interaction with many codes possible





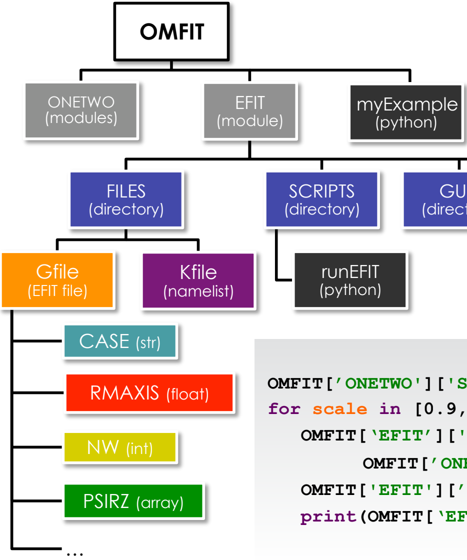
Data is organized in a tree structure which provides unified data access (similar to a file-system or MDS+)



- Hierarchical organization is self-descriptive
- Objects are grouped into modules: collections of objects used to solve elementary physics problem

- Files are interpreted and data populates the tree (namelists, NetCDF, matlab .mat, IDL .sav, ...)
- Objects can be accessed/manipulated in a unified way, independently of their type or origin
  e.g. **OMFIT**[“**EFIT**”][“**FILES**”][“**Gfile**”][“**NW**”]
- Objects have different capabilities depending on their types: scripts can be executed, files can be saved, arrays can be plotted, ...

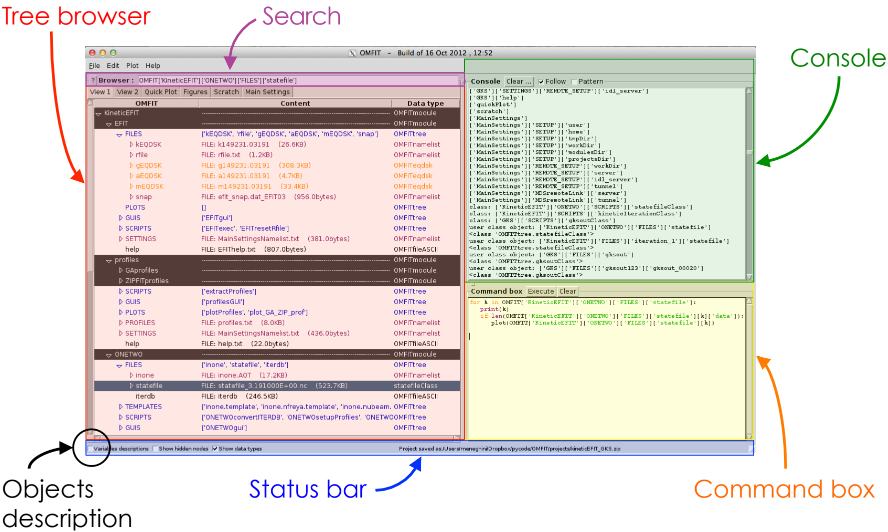
Unified data structure defines a memory space where tasks communication can dynamically occur



- Components and workflows are built with user-level Python scripts
- Scripts can call each other to build complex workflows

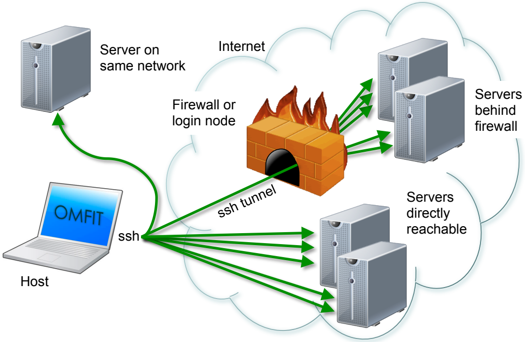
```
OMFIT['ONETWO']['SCRIPTS']['ONETWOexec'].run()
for scale in [0.9,1.0,1.1]:
    OMFIT['EFIT']['FILES']['Kfile']['IN1']['PRESSR']=\
          OMFIT['ONETWO']['FILES']['statefile']['press']*scale
    OMFIT['EFIT']['SCRIPTS']['EFITexec'].run()
    print(OMFIT['EFIT']['FILES']['Gfile']['RMAXIS'])
```

Top-level GUI to interactively manage the tree structure, execute and edit scripts and manipulate and visualize data

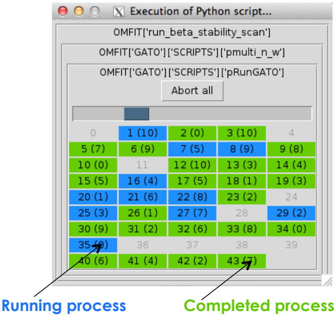


Easy to execute tasks remotely and in parallel with high level APIs

- Seamless execute codes and and manage files remotely
  - Let codes run codes where they already work!
  - Machine running OMFIT directs and stores data in OMFIT tree
- Parallel execution of the same task with different input parameters, on multiple remote machines
- Real-time monitoring of local / remote and serial / parallel tasks



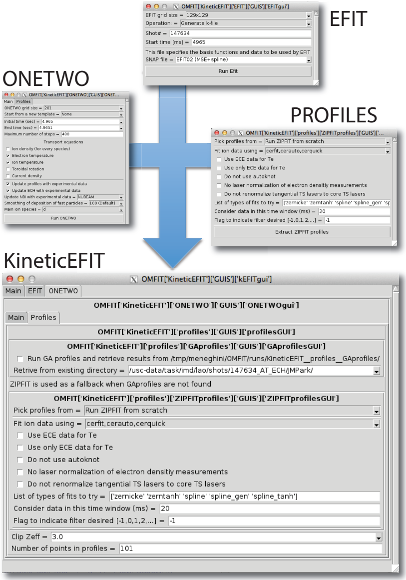
Monitor progress of parallel execution



Easy to create Graphical Users Interfaces (GUIs) with high level APIs

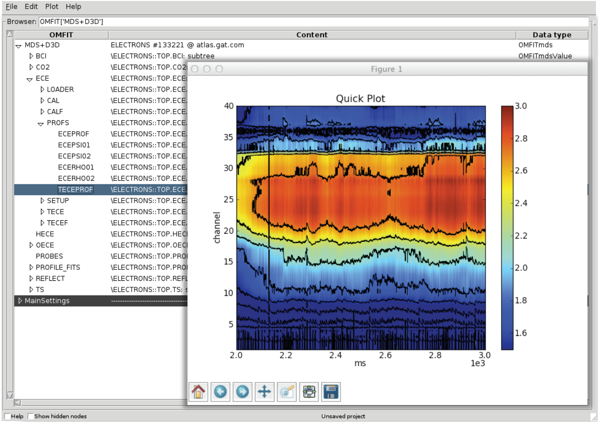
User GUIs speed-up routine analysis and hide many of the underlying complexities to inexperienced users

- GUIs are python scripts and are created by users themselves
- Quick and easy! For each GUI entry need to specify the OMFIT tree location associated with it
- GUIs can be nested to create comprehensive GUIs, while ensuring consistency

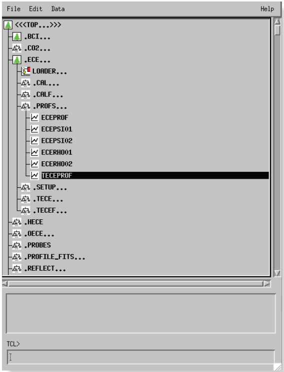


Directly access experimental data from the OMFIT tree

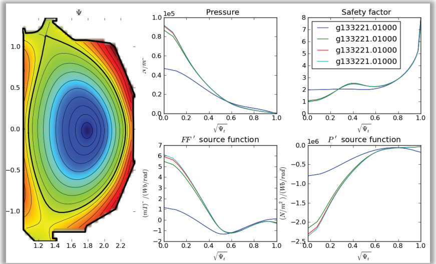
- Browse, search, plot and manipulate MDS+ data, SQL tables
- Creation of codes inputs: profiles, power, angles,..
- Validation: compare modeling results with experiments



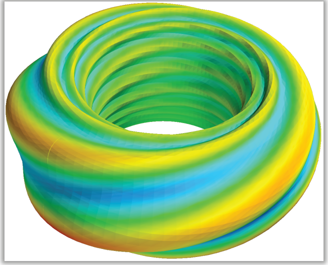
MDS+ traverser



Quickly visualize data in the OMFIT tree or create publication quality graphics with Python scripts



Kinetic EFIT iterations



M3D-C1 simulation of RMP pressure perturbation

1D/2D arrays are (over)-plotted with the push of a button

- Inspect inputs/outputs of different analyses / codes / iterations / ...
- Plots are interactive and can be customized (à la MATLAB)

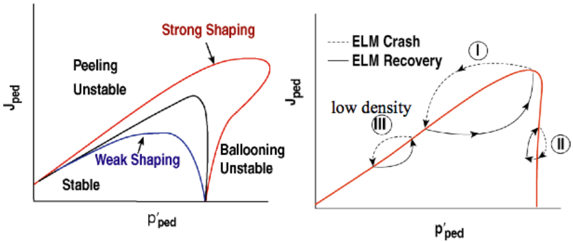
More sophisticated plots are scripted in Python

- Matplotlib library very similar to MATLAB and IDL plot commands
- Plotting scripts can be assigned to specific objects

Outline

1. Why OMFIT is a unique integrated modeling framework
2. Integrated analysis of edge stability experiments in DIII-D
3. More examples of OMFIT integrated analyses and modeling
4. Conclusions and future work

The Peeling-Ballooning model for edge stability and ELMs



- ELMs caused by intermediate n $(\sim 3 - 30)$ MHD instabilities
- Both ∇J and ∇P driven, with complex inter-dependencies:
  - Steep pressure gradient

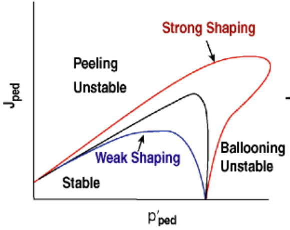
    DRIVE high n "ballooning" instabilities

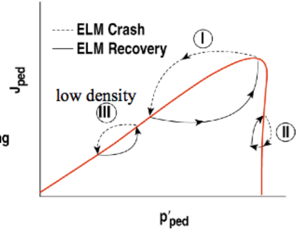
    STABILIZE "peeling" modes by increasing good curvature
  - High bootstrap current

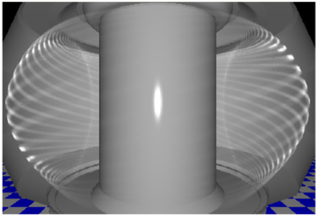
    DRIVE low n "peeling" instabilities

    STABILIZE "ballooning" modes by decreasing magnetic shear
- Limit-cycle around stability boundary can explain wide range of ELM phenomena observed in tokamaks

ELITE* code is the workhorse for DIII-D edge stability analysis







ELITE, n=18 mode structure

- ELITE is a 2D eigenvalue code, based on ideal MHD
  Generalization of ballooning theory:
  1. Incorporate surface terms which drive peeling modes
  2. Retain first two orders in $1/n$ stability (treats intermediate $n >\sim$5)

- Several steps are required to obtain an accurate ELITE analysis:
  1. Start from plasma equilibrium and kinetic profiles
     - Special attention to the edge pressure and current!
  2. Parametric variations of the pedestal pressure and current
  3. Run ELITE for ∇P and ∇J variations and for multiple n

* P. Snyder et al. Phys. Plasmas, Vol. 9, No. 5, May 2002

Kinetic equilibrium reconstructions are the first step for an accurate transport and stability analysis

Accuracy of equilibrium that can be reconstructed increases with availability of information:

For boundary and global parameters:

- **Magnetics (Flux loops and magnetic probes)**
  + Plasma boundary, β_p, l_i and I_p

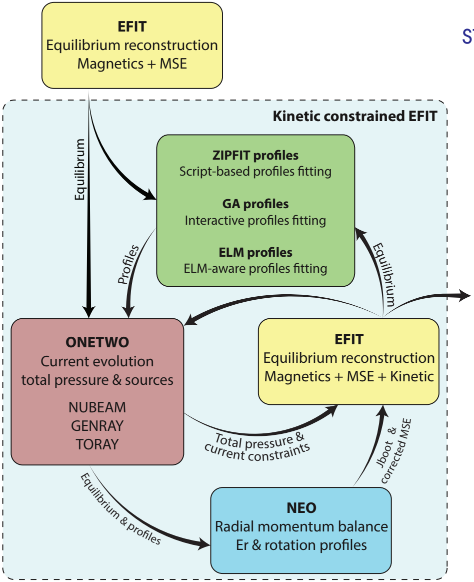
Full equilibrium reconstruction require:

- **Magnetics + MSE**
  + q profile $\rightarrow$ J profile
- **Magnetics + MSE + kinetic profiles**
  + Pressure profile and internal magnetic geometry

Physics models can also be used as constraints:

- **Fast particles pressure**
  - From NBI codes (Eg. NUBEAM, ...)
- **Current profile**
  - OH and bootstrap from neoclassical codes or Sauter model
  - RF & NBI from codes (Eg. TORAY, GENRAY, NUBEAM, ...)

Workflow of a DIII-D kinetic EFIT reconstruction in OMFIT



step **0** Run *magnetics + boundary + MSE* constrained EFIT

1.a Fit kinetic profiles in flux space (ZIPFIT, GAprofiles)

1.b Find p_{NBI} and J_{BS} running the ONETWO transport code

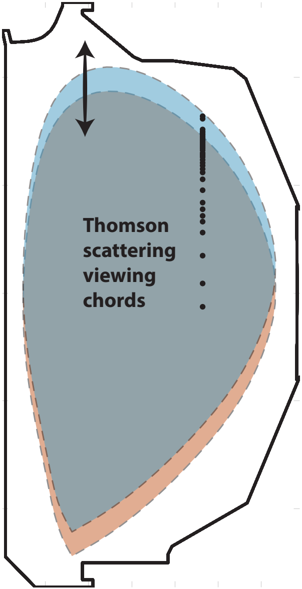
1.c Run *magnetics + boundary + MSE + kinetic* constrained EFIT

1.d Run *NEO* to get accurate predictions of J_{boot} and E_r

1.e Correct MSE data for Zeeman effect from E_r

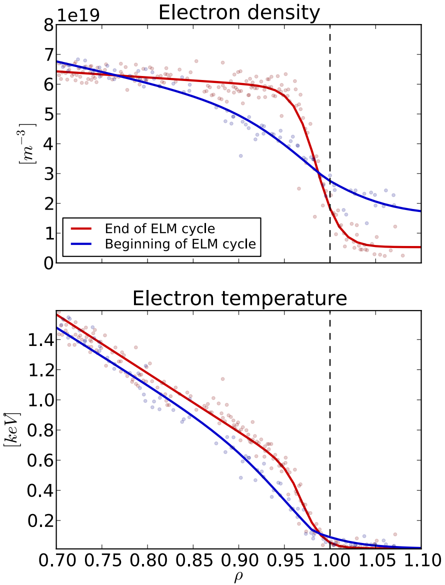
2...n Repeat **.a .b .c.d .e** with updated equilibrium

ELM-profile module in OMFIT allows accurate fitting of pedestal profiles as function of ELM cycle



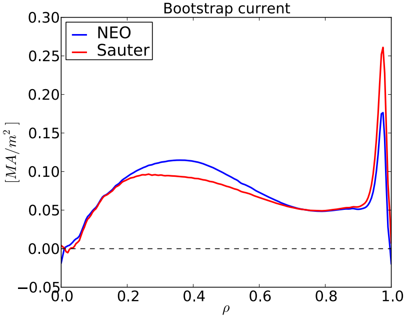
- In ELM stability experiments, Thomson scattering resolution is increased by sweeping plasma past the viewing chords
- Separatrix location is tracked based on magnetics-only EFIT reconstruction
- Data is binned as a function of D_α light emission $\rightarrow$ proxy for ELM cycle

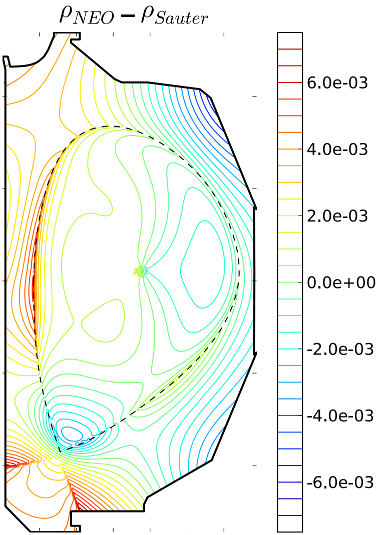
ELM-profile module in OMFIT allows accurate fitting of pedestal profiles as function of ELM cycle



- In ELM stability experiments, Thomson scattering resolution is increased by sweeping plasma past the viewing chords
- Separatrix location is tracked based on magnetics-only EFIT reconstruction
- Data is binned as a function of D_α light emission $\rightarrow$ proxy for ELM cycle

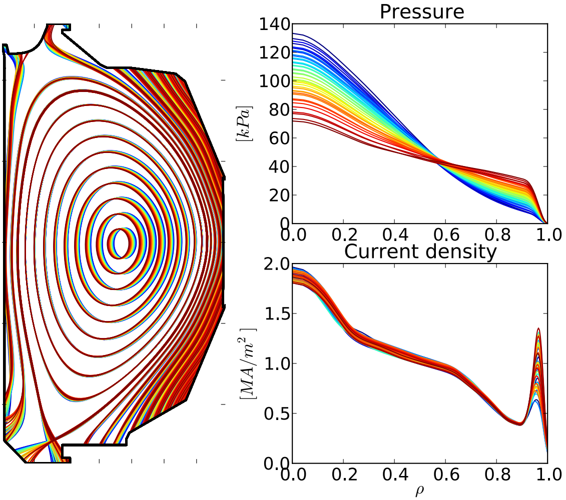
Getting accurate bootstrap current with NEO





- Sauter model accurate for most DIII-D cases
- In high collisionality cases, Sauter model can be off by as much as 40% from neoclassical calculations (e.g. from NEO)

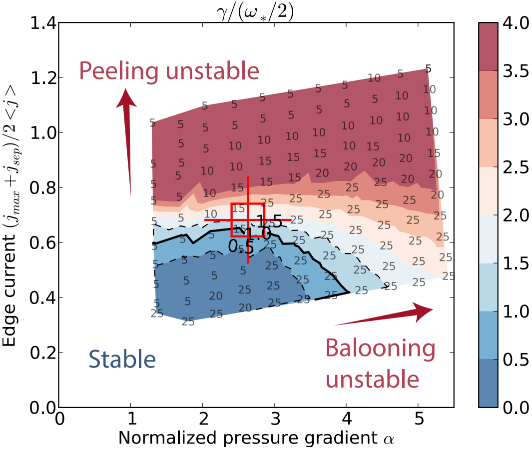
Parametric independent variations of the pedestal pressure and current with VARYPED tool



Uses T. Osborne's VARYPED tool perform scan of ∇P and ∇J in the pedestal:

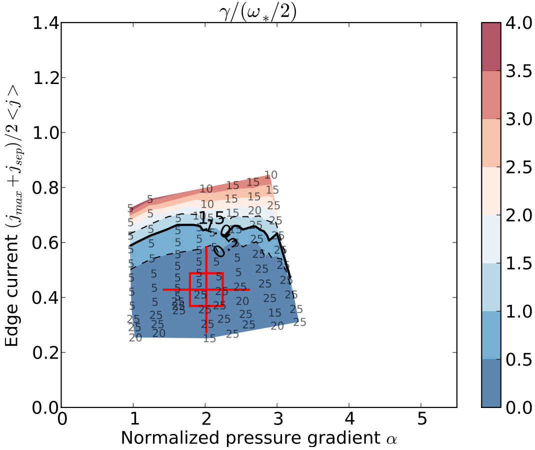
- constant stored energy
- constant total current
- fixed collisionality profile

Edge stability sensitivity analysis with ELITE
$\rightarrow$ ELM I H-mode (90-100% bin)



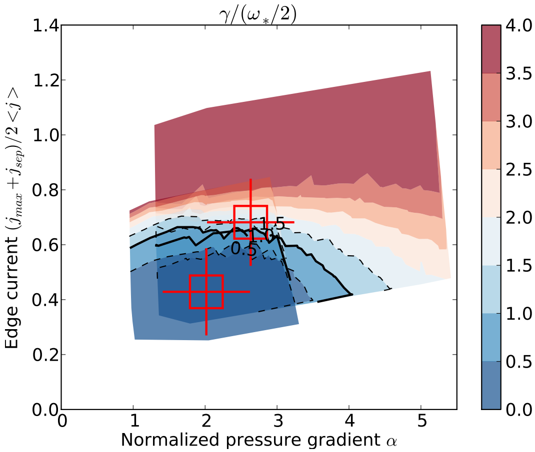
- Color represents growth rate of most unstable mode (numbered)
- Last ELM phase is at the limit of the PB stability

Edge stability sensitivity analysis with ELITE
$\rightarrow$ ELM I H-mode (60-70% bin)



- Color represents growth rate of most unstable mode (numbered)
- Last ELM phase is at the limit of the PB stability
- Earlier ELM phases are more and more stable

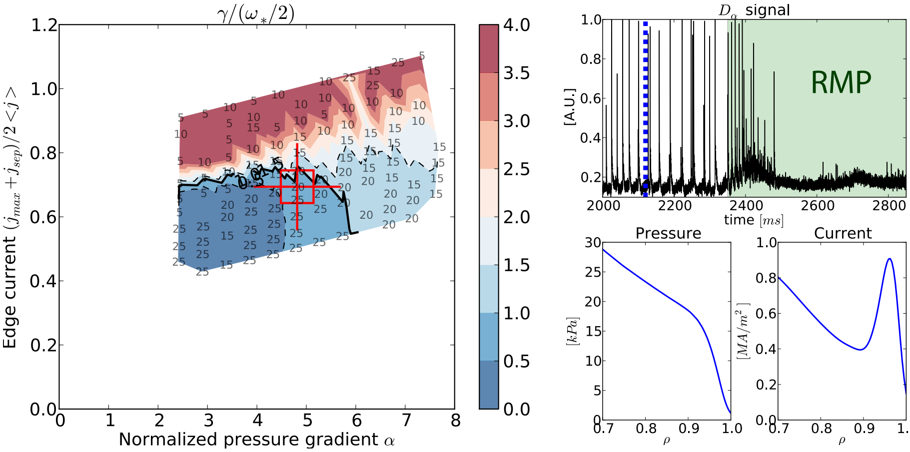
Edge stability sensitivity analysis with ELITE
$\rightarrow$ ELM I H-mode (90-100% bin & 60-70% bin)



- Color represents growth rate of most unstable mode (numbered)
- Last ELM phase is at the limit of the PB stability
- Earlier ELM phases are more and more stable
- Superposition between ELM phase scans shows good overlapping

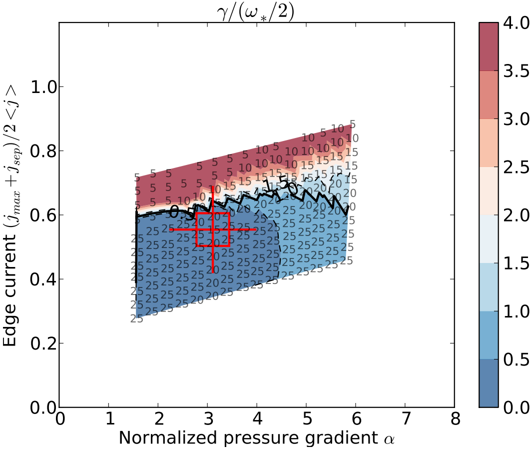
Edge stability sensitivity analysis with ELITE $\rightarrow$ RMP ELM suppressed H-mode (before RMP)

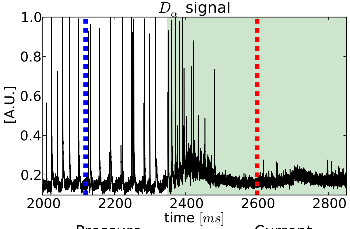
- Before RMP, ELMS are observed in the experiment
- 90-100% ELM phase profiles are at stability limit

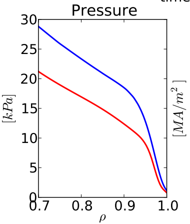


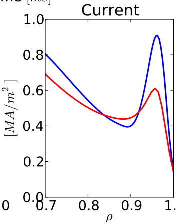
Edge stability sensitivity analysis with ELITE
→ RMP ELM suppressed H-mode (during RMP)

- After RMP, ELMS are suppressed in the experiment
- RMP profiles are in stable region









Outline

1. Why OMFIT is a unique integrated modeling framework
2. Integrated analysis of edge stability experiments in DIII-D
3. More examples of OMFIT integrated analyses and modeling
4. Conclusions and future work

OMFIT is routinely used to perform a wide range of integrated modeling studies and analyses

Equilibrium

EFIT

KineticEFIT

VaryPed

Transport

ONETWO

GCNMP

TGYRO

Gyro-kinetic

GYRO

TGLF

GKS

MHD stability

PEST3

GATO

Others

GENRAY

TORBEAM

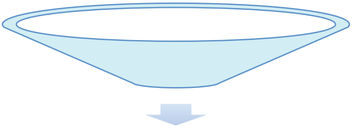
NUBEAM

M3DC1

NTV

Mag. flutter

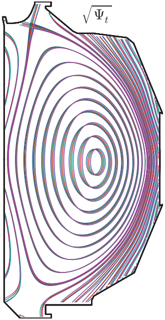
Exp. profiles

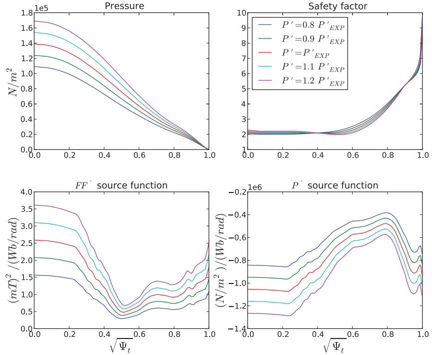


- OMFIT provides an ever-increasing list of ever-improving modules
- In general it is easy to support new codes, especially if they use standard file formats like *FORTRAN namelist* or *NetCDF*
- Users can integrate modules to create arbitrarily complex workflows
  - multi-dimensional parametric scans
  - iteration loops
  - non-linear optimization schemes
  - ...

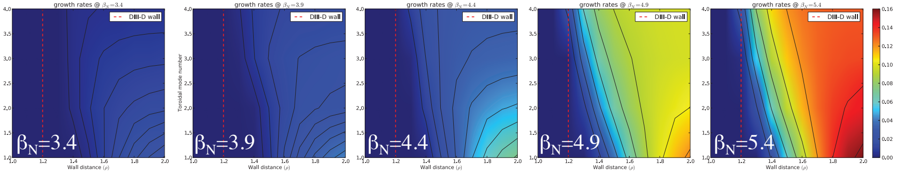
Survey of ideal MHD stability at increased β_n with GATO

Pressure scanned by scaling of P' and ideal MHD stability evaluated for different toroidal mode numbers n and wall distances (conformal wall)



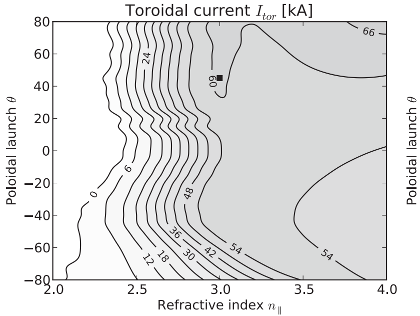


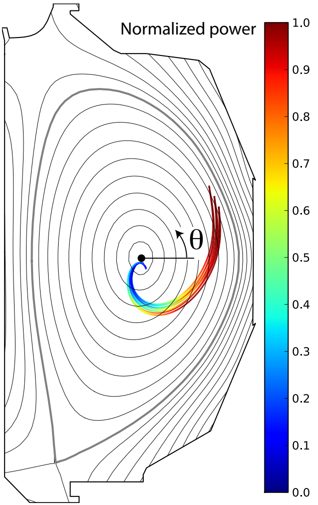
220 GATO simulations run 20 at a time in parallel on 3 different remote machines



Evaluation of whistler waves (also known as ‘helicons‘) current drive efficiency and location with GENRAY

- DIII-D target discharge #122976 with $\beta_n = 3.9$ (high β needed for absorption)
- Automated scan of launched $n_{\|}$ and poloidal angle θ of wave injection
- Target compares favorably $(60\ kA/MW)$ with respect to EC $(16\ kA/MW)$ and NBI $(26\ kA/MW)$





$500\ MHz \quad n_{\|} \sim 3.4 \quad \theta \sim 40\ deg$

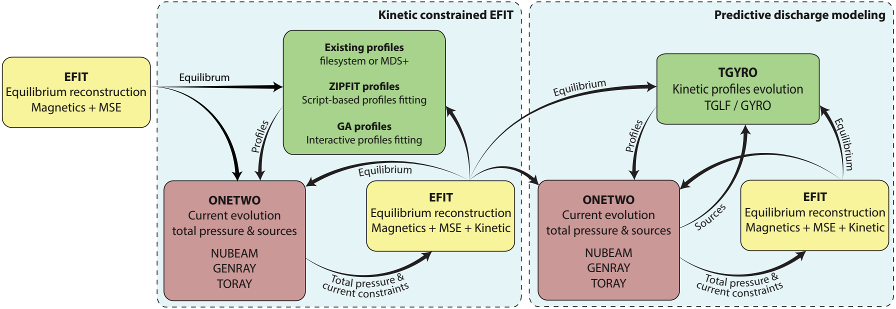
Extension of kinetic EFIT workflow for steady-state predictive modeling with TGYRO

Substitute: kinetic profiles **fitting** → kinetic profiles **prediction**

TGYRO efficiently solves the steady state transport equation:

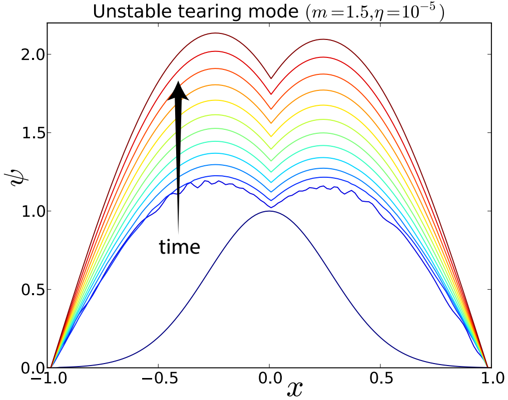
$$\Gamma_{neo}(x) + \Gamma_{turb}(x) = \Gamma_{target}(x) = \int_0^x V'(r)\, S(r)\, dr$$

- Neoclassical from NEO and turbulent from either TGLF or GYRO



Evolution of unstable tearing mode with BOUT++

- Preliminary integration of BOUT++ into OMFIT (runs on NERSC & GA workstations)
- BOUT++ Python tools easily embedded into OMFIT
- Can perform scans, optimization, interact with other modules



Example from O. Izacard model, run in OMFIT:

- Slab geometry
- Jensen equilibrium
- Gaussian initial condition
- BOUT++ growth rate compares well with analytic predictions

Outline

Conclusion

OMFIT is a framework for the every-day analysis and modeling needs of both theorists and experimentalist!

- **Tree data structure** provides unifying way to easily exchange data among codes and execute them in complicated workflows
- **Graphical environment** allows interactive analyses and inspection of intermediate results
- **Modular approach** and **collaborative environment** enable code reuse, promoting robust software and accelerated development
- **User-level GUIs** hide underlying complexities and facilitate streamlined analyses
- **Users retain full access** to input/output files, Python scripting
- **Powerful APIs** allow remote codes execution, reuse of existing scripts and widgets (IDL, matlab, shell, ...), access experimental data, GUI

Tutorials and more at *github.com/OMFIT/OMFITpublicData/wiki*

Future work

Integration with BOUT++ and OMFIT for automation of routine analyses (e.g. ELM analysis on DIII-D):

- Collect experimental data
- Mesh generation: EFIT $\rightarrow$ CORSICA $\rightarrow$ BOUT++
- Edit $\rightarrow$ compile $\rightarrow$ execute $\rightarrow$ collect data
- GUI for editing common parameters
- Post-processing (synthetic diagnostics?) and data analysis

More upcoming upgrades, including:

- Management of **batch queues on HPC** systems
- Integration with **EPED** for self consistent BC in transport simulations
- Integration with **SWIM** project $\rightarrow$ TORIC, AORSA, CQL3D, TLC, ...